```
In [1]: %runfile D:/GEN_AI_LAB/LAB1/EXP1.py --wdir
Here are a few ways to convert "She no went to the market" to standard English, depending on the desired nuance:

*   **She did not go to the market.** (Most direct and formal)
*   **She didn't go to the market.** (Common and natural with a contraction)
```

Date

Expt. No. 01 Page No. 01

Aim :

Exercises using Google GenAI for Text Correction, Classification, and Summarization

1A. Convert Ungrammatical Statements into Standard English.

Text Summarisation

→ Problem Statement : Correct the Grammatical mistakes in the sentence.

```
import os # used to interact with system environment variables
import google.genai as genai # Google GenAI library to access
 generative AI models
# os.environ ['GOOGLE_API_KEY'] = 'YOUR_API_KEY'
#↑ Temporarily sets the API key (already set as system environment
 variable)
client = genai.Client (api_key = "AIzaSyCoyr3sBIlSCbir60H8.....")
# Initializes the Google GenAI client to communicate with API.
# Assigning the model name for content generation
model = "gemini-2.5-flash" # Model used for natural language
 generation tasks
response = client.models.generate_content (model = model,
contents = "You will be provided with statements, and your task
 is to convert them to standard E "Here is the statement :
 She ~~xx~~ no went to the market.")# generates content from prompt
print (response.text) # Retrieves the generated text response
 from model
```

Teacher's Signature

```
In [2]: %runfile D:/GEN_AI_LAB/LAB1/EXP1.2.py --wdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760770592.397108   18468 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is not enabled.
Here's a categorization of the companies based on their primary industry sectors:

**1. Technology Sector:**
*   **Microsoft Corporation:** Software, Cloud Computing, Hardware (e.g., Windows, Azure, Xbox)
*   **Apple Inc:** Consumer Electronics, Software, Online Services (e.g., iPhone, Mac, App Store)
*   **Amazon.com, Inc:** E-commerce, Cloud Computing, Digital Streaming, Artificial Intelligence

**2. Pharmaceuticals & Healthcare Sector:**
*   **Roche Holding AG:** Pharmaceuticals and Diagnostics
*   **Pfizer Inc:** Pharmaceuticals
*   **Johnson & Johnson:** Pharmaceuticals, Medical Devices, Consumer Health Products

**3. Financial Services / Banking Sector:**
*   **JPMorgan Chase & Co.:** Investment Banking, Retail Banking, Asset Management
*   **Bank of America Corporation:** Retail Banking, Commercial Banking, Investment Banking
*   **Industrial and Commercial Bank of China (ICBC):** Commercial Banking, Personal Banking, Corporate Banking, Investment Banking
```

1B. Text Classification

Problem Statement : Classify the companies given below:
Microsoft Corporation, Roche Holding AG, Apple Inc, Amazon.com, Inc, Pfizer Inc, JPMorgan Chase & Co., Johnson & Johnson, Bank of America Corporation, Industrial and Commercial Bank of China.

```
import google.generative as genai
# Configures API access using the API key stored in environment variable
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])
# Initializes the specific AI model for text generation
model = genai.GenerativeModel('gemini-2.5-flash')
# Prompt to classify companies based on their sectors or types
prompt = """
Categorize the following companies :
Microsoft Corporation, Roche Holding AG, Apple Inc, Amazon.com,
Inc, Pfizer Inc, JPMorgan Chase & Co., Johnson & Johnson,
Bank of America Corporation, Industrial and Commercial
Bank of China."""
# Sends the prompt to the model and gets generated text output
response = model.generate_content(prompt)
# Extracts only the text portion of the generated output
print(response.text)
```

Ex2: Text Summarisation

Problem Statement : Summarise the minutes of the meeting given below in 4 bullet points :

Car Manufacturing Company

Board of Directors Meeting Minutes

Agenda :

1. Call to Order
2. Approval of previous meeting minutes
3. CEO's Report
4. Financial Report

Meeting Minutes :

1. Call to Order :

- The meeting was called to order by Bharath Thippireddy.

2. Approval of Previous Meeting Minutes :

- The minutes of the previous board meeting, held on [Date], were reviewed and approved.

3. CEO's Report :

- Bharath Thippireddy presented an overview of the company's performance, highlighting key achievements and challenges. Key points discussed include :
  - Sales figures for the last quarter.
  - Progress on cost reduction initiatives.
  - Highlights from recent industry events.
  - The CEO answered questions from board members.

4. Financial Report :

- The Chief Financial Officer ([CFO's Name]) presented the financial report. Key financial metrics discussed include.

- Revenue and profit margins.
- Budget vs. actual expenditure.
- Cash flow analysis.
- The board discussed financial performance and the impact on shareholder value.

```
import google.generativeai as genai
# genai.configure() → Reconfigures API to make sure the model can access key
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])
# Loads the generative model to perform summarization
model = genai.GenerativeModel('gemini-2.5-flash')
prompt = """
Summarise the minutes of the meeting given below in 4 numbered
bullets also include names
Car Manufacturing Company
Board of Directors Meeting Minutes
Agenda:
1. Call to Order
2. Approval of Previous Meeting Minutes
3. CEO's Report
4. Financial Report
Meeting Minutes:
1. Call to Order:
- The meeting was called to order by Bharath Thippireddy.
2. Approval of Previous Meeting Minutes:
- The minutes of the previous board meeting, held on [Date],
were reviewed and approved.
3. CEO's Report:
- Bharath Thippireddy presented an overview of the
```

```
In [1]: %runfile D:/GEN_AI_LAB/LAB1/EXP1.3.py --wdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760771322.566234   10112 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP
and untrusted ALTS is not enabled.
Here is a summary of the meeting minutes:

1.  The meeting was called to order by Bharath Thippireddy.
2.  The minutes from the previous board meeting were reviewed and subsequently approved.
3.  Bharath Thippireddy, the CEO, delivered a report covering the company's performance, recent sales
figures, progress on cost reduction, and highlights from industry events, also addressing questions
from board members.
4.  The Chief Financial Officer, Kushal v, presented the financial report, detailing revenue, profit
margins, budget vs. actual expenditure, and cash flow analysis, which the board then discussed in
relation to shareholder value.
```

company's performance, highlighting key achievements and challenges.

Key points discussed include:

- Sales figures for the last quarter.
- Progress on cost reduction initiatives.
- Highlights from recent industry events.
- The CEO answered questions from board members.

4. Financial Report:

- The Chief Financial Officer (Kushal V) presented the financial report. Key financial metrics
- Revenue and profit margins.
- Budget vs. actual expenditure.
- Cash flow analysis.
- The board discussed financial performance and the impact on shareholder value """.

```
# Generates a summarized version of provided meeting minutes
response = model.generate_content(prompt)

# Displays the summarized result generated by AI model
print(response.text)
```

Ex 3: Sentiment Analysis

Problem statement : To develop a Python program using Google Generative AI that analyzes the sentiment of given social media posts and classifies each post as positive, negative, or neutral, providing a brief explanation for each classification.

```
import google.generativeai as genai # lib to use GoogleGenAI models
import os # used to access system environment variables like API keys.
# Configure the API key for authentication
# genai.configure() → connects the program to the Google GenAI servic
genai.configure (api_key = os.environ ["GOOGLE_API_KEY"])
# Loads the specific Gemini model to perform text analysis or generation.
model = genai.GenerativeModel ('gemini-2.5-flash')
# The mode will read each post and classify its sentiment as positive,
negative, or neutral.
prompt = """ "text": Below are several social media posts. For
each post, please analyze the sentiment expressed and
classify it as either positive, negative, or neutral. Provide
a brief explanation for your classification.
1. Just got my hands on the new xphone - absolutely loving the
camera and battery life! #TechLove
2. Disappointed with the XPhone. It's pricey and not much
different from the last model. Expected more #TechTalk
3. XPhone's latest update is a game-changer for mobile gaming.
The graphics are just incredible!
4. Can't believe I waited this long for the XPhone... it's
underwhelming and overpriced. Back to my old phone, I guess.
```

```
In [2]: %runfile D:/GEN_AI_LAB/LAB2/EXP2.1.py --wdir
E0000 00:00:1760771575.919944   10112 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP
and untrusted ALTS is not enabled.
Here's the sentiment analysis for each post:

1.  **Sentiment: Positive**
    *   **Explanation:** The user expresses strong positive feelings with phrases like "absolutely
loving the camera and battery life," indicating high satisfaction.

2.  **Sentiment: Negative**
    *   **Explanation:** The post clearly states "Disappointed," and lists negative points such as
being "pricey" and "not much different from the last model," along with unfulfilled expectations.

3.  **Sentiment: Positive**
    *   **Explanation:** The user uses highly positive language, calling the update a "game-changer"
and describing the graphics as "incredible," showing enthusiasm and satisfaction.

4.  **Sentiment: Negative**
    *   **Explanation:** The post conveys strong dissatisfaction through words like "underwhelming"
and "overpriced," culminating in the decision to go "Back to my old phone."

5.  **Sentiment: Positive**
    *   **Explanation:** The user states the XPhone "has exceeded my expectations" and highlights
positive attributes like "Fast, sleek," and "standout" AI features.
```

5. The XPhone has exceeded my expectations. Fast, sleek, and the new AI features are a standout! #Innovation """

sends the prompt to Gemini for processing and returns a detailed text response (classification with reasoning)

```
response.= model.generate_content (prompt)
```

extracts and displays only the textual part of the model's output

```
print (response.text)
```

```
In [1]: %run-file D:/GEN_AI_LAB/LAB2/EX02_2.py --mdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
I0000 00:00:1768/72809.483435   23832 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is not enabled.
Alright, my dear! As a fellow cake enthusiast, I'm thrilled to share some of my absolute favorite, tried-and-true cake recipes. Each one offers something a little different, ensuring you'll find the perfect
treat for any occasion. Let's get baking!

---


### 1. My Signature Classic Vanilla Birthday Cake

This is my go-to recipe for a light, fluffy, and utterly delicious vanilla cake that pairs perfectly with almost any frosting. It's a crowd-pleaser and always brings smiles!

**Yields:** One 2-layer 8-inch cake (approx. 12 servings)
**Prep time:** 30 minutes
**Bake time:** 25-30 minutes

**Ingredients:**

*   **For the Cake:**
    *   2 ¾ cups (300g) all-purpose flour, sifted
    *   1 ½ teaspoons baking powder
    *   ½ teaspoon salt
    *   1 cup (226g) unsalted butter, softened
    *   2 cups (400g) granulated sugar
    *   4 large eggs, at room temperature
    *   1 tablespoon pure vanilla extract (the good stuff!)
    *   1 cup (240ml) whole milk, at room temperature

*   **For the Simple Buttercream Frosting:**
    *   1 cup (226g) unsalted butter, softened
    *   4 cups (450g) powdered sugar, sifted
    *   2 teaspoons pure vanilla extract
    *   2-4 tablespoons heavy cream or milk, as needed
    *   Pinch of salt

**Equipment:** Two 8-inch round cake pans, parchment paper, electric mixer
```

```
**Instructions:**

1.  **Prep:** Preheat your oven to 350°F (175°C). Grease and flour your two 8-inch round cake pans, then line the bottoms with parchment paper.
2.  **Combine Dry Ingredients:** In a medium bowl, whisk together the sifted flour, baking powder, and salt. Set aside.
3.  **Cream Butter & Sugar:** In a large bowl with an electric mixer (or stand mixer with paddle attachment), cream the softened butter and granulated sugar on medium-high speed until light and fluffy, about
3-5 minutes. Scrape down the sides of the bowl.
4.  **Add Eggs & Vanilla:** Add the eggs one at a time, beating well after each addition. Mix in the vanilla extract.
5.  **Alternate Wet & Dry:** With the mixer on low speed, gradually add the dry ingredients to the butter mixture in three additions, alternating with the milk in two additions. Begin and end with the dry
ingredients (e.g., 1/3 dry, 1/2 milk, 1/3 dry, 1/2 milk, 1/3 dry). Mix *just* until combined. Do not overmix!
6.  **Bake:** Divide the batter evenly between the prepared cake pans. Bake for 25-30 minutes, or until a wooden skewer inserted into the center comes out clean.
7.  **Cool:** Let the cakes cool in the pans for 10-15 minutes before inverting them onto a wire rack to cool completely.
8.  **Make Frosting:** While cakes cool, make the frosting. In a large bowl, beat the softened butter until creamy. Gradually add the sifted powdered sugar, vanilla extract, and salt. Start with 2 tablespoons
of cream, then add more as needed, beating until light and fluffy.
9.  **Assemble:** Once the cakes are completely cool, level them if necessary. Place one layer on your serving plate, spread with a generous layer of frosting, top with the second cake layer, and then frost
the top and sides of the entire cake. Decorate as desired!

---


### 2. Rich & Velvety Chocolate Fudge Cake

For the chocolate lovers, this cake is pure indulgence. It's incredibly moist, deeply chocolatey, and has a tender crumb that melts in your mouth.

**Yields:** One 2-layer 8-inch cake (approx. 12 servings)
**Prep time:** 25 minutes
**Bake time:** 30-35 minutes

**Ingredients:**

*   **For the Cake:**
    *   2 cups (240g) all-purpose flour, sifted
    *   ¾ cup (75g) unsweetened cocoa powder (Dutch-processed for darker color, natural for brighter)
    *   1 ½ teaspoons baking soda
    *   1 teaspoon salt
    *   2 cups (400g) granulated sugar
    *   1 cup (240ml) buttermilk, at room temperature (or 1 cup milk + 1 tbsp white vinegar, let sit 5 mins)
    *   ½ cup (120ml) vegetable oil
    *   2 large eggs, at room temperature
    *   1 teaspoon pure vanilla extract
    *   1 cup (240ml) hot strong coffee (or hot water)
```

```
*   **For the Chocolate Fudge Frosting:**
    *   1 cup (226g) unsalted butter, softened
    *   1 cup (100g) unsweetened cocoa powder
    *   4 cups (450g) powdered sugar, sifted
    *   ½ cup (120ml) milk or heavy cream
    *   1 teaspoon pure vanilla extract
    *   Pinch of salt

**Equipment:** Two 8-inch round cake pans, parchment paper, electric mixer

**Instructions:**

1.  **Prep:** Preheat your oven to 350°F (175°C). Grease and flour your two 8-inch round cake pans, then line the bottoms with parchment paper.
2.  **Combine Dry Ingredients:** In a large bowl, whisk together the sifted flour, cocoa powder, baking soda, salt, and granulated sugar.
3.  **Add Wet Ingredients (Except Coffee):** Add the buttermilk, vegetable oil, eggs, and vanilla extract to the dry ingredients. Beat with an electric mixer on medium speed until well combined, scraping down
the sides of the bowl.
4.  **Add Hot Coffee:** Slowly pour in the hot coffee (or hot water) and mix on low speed until the batter is smooth. The batter will be quite thin - this is normal and makes it incredibly moist!
5.  **Bake:** Divide the batter evenly between the prepared cake pans. Bake for 30-35 minutes, or until a wooden skewer inserted into the center comes out clean.
6.  **Cool:** Let the cakes cool in the pans for 10-15 minutes before inverting them onto a wire rack to cool completely.
7.  **Make Frosting:** While cakes cool, make the frosting. In a large bowl, beat the softened butter until creamy. Gradually add the cocoa powder and sifted powdered sugar. Add the milk/cream, vanilla, and
salt. Beat until smooth, rich, and spreadable. If it's too thick, add a bit more milk; if too thin, add more powdered sugar.
8.  **Assemble:** Once the cakes are completely cool, level them if necessary. Place one layer on your serving plate, spread with a generous layer of fudge frosting, top with the second cake layer, and then
frost the top and sides of the entire cake.

---


### 3. Bright & Zesty Lemon Drizzle Loaf Cake

This lemon drizzle cake is a ray of sunshine! It's wonderfully moist, packed with fresh lemon flavor, and the sugary drizzle creates a delightful tangy crust. Perfect for afternoon tea or a light dessert.

**Yields:** One 9x5 inch loaf cake (approx. 8-10 slices)
**Prep time:** 20 minutes
**Bake time:** 45-55 minutes
```

Ex4: Generating Multiple Responses

```
# Import the Google Generative AI library
import google.generativeai as genai
# impor the os module to access environment variables
import os
# The API key is retrieved from an environment variable name
  "GOOGLE_API_KEY"
genai.configure(api_key = os.environ["GOOGLE_API_KEY"])
# Load the gemini 2.5 flash model to generate AI content
model = genai.GenerativeModel('gemini-2.5-flash')
# Define input prompt
prompt = """ "text": "You are a cake baker"
             "text": "Get me 3 recipe to bake a cake" """
# Generates response based on the given prompt
response = model.generate_content(prompt)
# Print only the text part of the model's output
print(response.text)
```

Ingredients:

- **For the Cake:**
  - 1 ¾ cups (210g) all-purpose flour
  - 2 teaspoons baking powder
  - ½ teaspoon salt
  - ½ cup (113g) unsalted butter, softened
  - 1 cup (200g) granulated sugar
  - 3 large eggs, at room temperature
  - Zest of 2 large lemons
  - ¼ cup (60ml) fresh lemon juice
  - ½ cup (120ml) whole milk or plain Greek yogurt, at room temperature
- **For the Lemon Drizzle:**
  - ½ cup (60g) powdered sugar
  - 2-3 tablespoons fresh lemon juice

Equipment: One 9x5 inch loaf pan, parchment paper, electric mixer.

Instructions:

1. **Prep:** Preheat your oven to 350°F (175°C). Grease and flour a 9x5 inch loaf pan, then line the bottom with parchment paper, leaving an overhang on the long sides to help lift the cake out.
2. **Combine Dry Ingredients:** In a medium bowl, whisk together the flour, baking powder, and salt. Set aside.
3. **Cream Butter & Sugar:** In a large bowl with an electric mixer, cream the softened butter and granulated sugar until light and fluffy, about 3-4 minutes.
4. **Add Eggs & Zest:** Add the eggs one at a time, beating well after each addition. Beat in the lemon zest.
5. **Alternate Wet & Dry:** With the mixer on low speed, gradually add the dry ingredients to the butter mixture in three additions, alternating with the combined lemon juice and milk/yogurt in two additions. Begin and end with the dry ingredients. Mix *just* until combined.
6. **Bake:** Pour the batter into the prepared loaf pan and smooth the top. Bake for 45-55 minutes, or until a wooden skewer inserted into the center comes out clean. If the top browns too quickly, tent loosely with foil.
7. **Make Drizzle (While Cake Bakes):** In a small bowl, whisk together the powdered sugar and lemon juice until smooth. It should be thick but pourable.
8. **Drizzle & Cool:** As soon as the cake comes out of the oven, while it's still hot and in the pan, pour the lemon drizzle evenly over the top. The hot cake will absorb the tangy goodness.
9. **Cool Completely:** Let the cake cool in the pan for about 20 minutes, then use the parchment paper overhang to lift it onto a wire rack to cool completely before slicing and serving.

A Baker's Top Tips for Success:

- **Room Temperature Ingredients:** This is crucial! Butter, eggs, and milk at room temperature emulsify better, creating a smoother, more uniform batter and a finer crumb.
- **Don't Overmix:** Overmixing develops the gluten in the flour, leading to a tough, dry cake. Mix *just* until the ingredients are combined.
- **Accurate Measuring:** Especially with flour, spoon it into your measuring cup and level it off with a straight edge (don't scoop directly from the bag). Better yet, use a kitchen scale for precision!
- **Preheat Oven Properly:** Give your oven at least 15-20 minutes to come up to temperature and stabilize before putting in your cake.
- **Pans & Parchment:** Always grease, flour, and line your cake pans. It's a small step that saves a lot of heartache when unmolding!
- **Cool Completely:** Frosting a warm cake is a recipe for disaster. Be patient and let your cakes cool entirely before frosting.

Happy baking, my friend! I hope you enjoy these recipes as much as I do. Let me know which one you try first!

```
In [1]: %runfile D:/GEN_AI_LAB/LAB2/EXP2.3.py --wdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760776847.087711   25572 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP
and untrusted ALTS is not enabled.
This Python code prompts the user for an integer and then determines and prints whether that number is
prime or not.
```

Ex5: Code Explanation

```
import google.generatiai as genai #Import GoogleGenAI library
import os # to access environment variables
genai.configure(api_key =os.environ["GOOGLE_API_KEY"])
# Configuring the Generative AI client using your Google API key.

#Load the Gemini-2.5-flash model
model = genai. GenerativeModel ('gemini-2.5-flash')

# Define prompt to explain what a given code does in sentence
prompt = """ Explain what this python code does :

num = int(input("Enter a number :"))
primeFlag = True
for i in range (2, number):
    if number % i == 0:
        primeFlag = False
        break
if primeFlag == True :
    print (num, " is prime.")
else :
    print (num, "is not prime.")
"""

# Generate AI model's response based on prompt
response = model. generate_content (prompt)
# print only the text output from the model's response
print (response. text)
```

```
In [1]: %runfile D:/GEN_AI_LAB/LAB2/EXP2.4.py --wdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760789706.149676   24668 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
OpenAI is an **artificial intelligence (AI) research and deployment company** whose primary mission is to ensure that
artificial general intelligence (AGI) benefits all of humanity.

Here's a breakdown of what that means:

1.  **AI Research & Development:** They are at the forefront of developing highly advanced AI models. Their work involves
cutting-edge research in machine learning, neural networks, and various AI subfields.

2.  **Goal: Artificial General Intelligence (AGI):** Their ultimate aim is to build AGI - highly autonomous systems that
outperform humans at most economically valuable work. They believe this technology could be immensely powerful and
transformative.

3.  **Key Products & Innovations:**
    *   **ChatGPT:** Their most well-known product, a conversational AI chatbot capable of understanding and generating
human-like text, answering questions, writing essays, coding, and more.
    *   **GPT Models (Generative Pre-trained Transformers):** A series of powerful language models (like GPT-3, GPT-4) that
underpin ChatGPT and many other applications, capable of text generation, summarization, translation, and various natural
language processing tasks.
    *   **DALL-E:** An AI system that can generate realistic and imaginative images from text descriptions.
    *   **Whisper:** An open-source automatic speech recognition (ASR) system.
    *   **API Platform:** They offer access to their AI models via an API, allowing developers and businesses to integrate
powerful AI capabilities into their own applications.

4.  **Organizational Structure (Important Nuance):**
    *   **Founded:** In 2015, it was founded as a non-profit research lab by a group including Elon Musk, Sam Altman, Greg
Brockman, and others, with a significant initial investment. The "Open" in its name originally emphasized a commitment to
open research and collaboration.
    *   **Restructuring:** In 2019, they restructured, creating a "capped-profit" subsidiary under the original non-profit.
This allowed them to raise massive amounts of capital (e.g., from Microsoft) to fund the immense computational and talent
costs of large-scale AI research, while still attempting to maintain a mission-driven approach where profits for investors
are capped, and the non-profit parent retains control over the mission and governance.

5.  **Impact and Focus:** OpenAI has played a pivotal role in accelerating the mainstream adoption and understanding of
generative AI. They are focused not only on developing powerful AI but also on the safety, ethics, and alignment of these
systems, aiming to ensure they are beneficial and not harmful to humanity.

In essence, OpenAI is a leading organization pushing the boundaries of AI, bringing powerful tools to the public and
developers, and grappling with the profound societal implications of their technology, particularly as they pursue the
development of AGI.
```

Ex:06 Error Handling

```
import os # to access environment variables
import google.generativeai as genai # import GoogleGenAI lib

from google.api_core import exceptions # for handling API-related errors

# Configure the Generative AI client using API key stored in
an environment variable
genai.configure(api_key = os.environ["GOOGLE_API_KEY"])

#Load the gemini 2.5 flash model
model = genai.GenerativeModel('gemini-2.5-flash')

#Use a try-except block to safely handle errors during API calls
try:
    # Generate AI content by asking the model a question.
    response = model.generate_content("What is OpenAI?")

    # Print the text part of AI's response
    print(response.text)

# Handle errors related to invalid or unathorized API keys
except (exceptions.InvalidArgument, exceptions.PermissionDenied) as e:
    print(f" API Key Error: {e}")
```

Hello there! I'd be happy to explain Pi to you. Think of me as your personal math guide for today!

What is Pi?

Let's start by thinking about circles. Every circle, big or small, has two main measurements we often talk about:

1. **Diameter:** This is the distance straight across the circle, passing right through its center. Imagine drawing a line from one side to the other, going through the bullseye.
2. **Circumference:** This is the distance all the way around the outside edge of the circle. If you were to walk along the very edge of a circular pond, that's the circumference.

Now, here's the cool part about Pi:

Pi (π) is a special number that tells us the relationship between a circle's circumference and its diameter.

No matter how big or small your circle is—whether it's a tiny button or a giant Ferris wheel—if you divide its circumference by its diameter, you *always* get the same number. That number is Pi!

Think of it like this: If you take the "length around" any circle (its circumference) and divide it by the "length across" that same circle (its diameter), you'll always end up with approximately **3.14159...**

So, mathematically, we can write it as:
$\pi = \text{Circumference} \div \text{Diameter}$

Here are a few more important things to know about Pi:

* **It's a Constant:** This means it's always the same value for every single circle in the universe. It doesn't change!
* **It's Irrational:** This is a fancy way of saying that its decimal goes on forever without repeating any pattern. So, 3.1415926535... keeps going and going! Because it's impossible to write down its exact value, we often use the Greek letter π as its symbol, or we approximate it as 3.14 or 22/7 for simpler calculations.
* **It's Everywhere:** Pi isn't just used for circles! It pops up in all sorts of places in math, physics, engineering, and even art. It's truly a fundamental number in the universe.

So, in short, Pi is that magical constant number that describes a fundamental property of all circles.

Summary of Pi:

* Pi (π) is the **constant ratio** of a circle's **circumference** (distance around) to its **diameter** (distance across). For any circle, Circumference $\div$ Diameter always equals Pi.
* It's an **irrational number** that starts with approximately 3.14159... and its decimal digits go on forever without repeating, making it a fundamental constant in mathematics and science.

Ex 7: Chatbot Application

```
import os # os module to access environment variables
import google.generativeai as genai # import Google GenerativeAI lib
# Configure the Generative AI client with your API key
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])
#model is fast & efficient generative model capable of understanding text
model = genai.GenerativeModel('gemini-2.5-flash')
#The model is told to act as Math tutor, explain concept of Pi
prompt = """
        You are a Math Tutor.
        First, explain what is Pi.
        After your explanation, summarize it in two bullet
        points."""
# Generate the response from the model based on the prompt
response = model.generate_content(prompt)
# print the text output generated by the model
print(response.text)
```

```
In [1]: %runfile D:/GEN_AI_LAB/LAB2/EXP2.6.py --wdir
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only
tokenizers, configuration and file/data utilities can be used.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760792630.628554   19480 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP
and untrusted ALTS is not enabled.
Here are some name suggestions for a sock company:

**Serious and Professional Name:**

*   **Stance Textiles**
    *   *Reasoning:* "Stance" implies quality, reliability, and a firm foundation, directly relating
to the act of standing and footwear. "Textiles" is a professional and accurate industry term, lending
an air of established craftsmanship.

---

**Creative and Fun Names (List of Five):**

1.  **Sock Hop**
    *   *Reasoning:* Evokes a retro, playful, and energetic feel, suggesting fun and movement.
2.  **Knit Happens**
    *   *Reasoning:* A clever, memorable pun that's lighthearted and hints at the product's creation.
3.  **Foot Candy**
    *   *Reasoning:* Suggests colorful, appealing, and desirable socks that are a treat for your feet.
4.  **Sole Mates**
    *   *Reasoning:* A charming pun that implies comfort, connection, and the perfect pair for your
feet.
5.  **Wiggle Toes Club**
    *   *Reasoning:* Playful and whimsical, it highlights the comfort and freedom socks should
provide, inviting customers into a fun community.
```

Ex:8 Prompts and LLMs Using Langchain

```
import os #used to access your API key from environment variables

#Import the ChatGoogleGenerativeAI class from Langchain's Google
 Generative AI integration.
# This class lets you use Gemini models through the LangChain framework
from langchain_google_genai import ChatGoogleGenerativeAI

llm = ChatGoogleGenerativeAI (model = "gemini-2.5-flash",
temperature = 0.8 )# temp →controls creativity (0-factual, 1-creative)
# Define the text prompt.
prompt = """Suggest names for a company that produces socks.
 First, provide one serious and professional name.
Then, provide a list of five creative and fun names."""

# Send the prompt to the Gemini model and get the response.
 response = llm. invoke (prompt)

# print the generated text content from the model
 print ( ~~the~~ response. content)
```

I. Nature-Inspired & Eco-Focused: (Emphasize sustainability and natural origin)

1. **BambooBloom:** Suggests growth, natural beauty, and the flourishing of a sustainable choice.
2. **TerraBrush:** "Terra" means Earth, clearly linking to the planet and eco-friendliness.
3. **Veridian Oral:** "Veridian" is a shade of green, symbolizing nature and vitality. "Oral" directly relates to the product.
4. **Root & Bristle:** Evokes the natural origin (roots of bamboo) and the product's function.
5. **Evergreen Smile:** "Evergreen" implies longevity and continuous green practices, connected to a healthy smile.
6. **The Green Shoot:** "Shoot" refers to new bamboo growth, symbolizing freshness and renewal.
7. **EcoGrove:** Combines "eco" with "grove" (a small forest of bamboo), painting a natural picture.
8. **PureStalk:** "Stalk" refers to the bamboo, and "pure" highlights the natural, clean aspect.

II. Modern & Catchy: (Short, memorable, and unique)

9. **Bamboom!** A playful and energetic twist on "bamboo," suggesting impact and growth.
10. **Bristle & Bloom:** Short, alliterative, and combines the product's part with natural growth.
11. **Sustainablu:** A creative blend of "sustainable" with a clean, modern sound (and "blue" for clear skies/water).
12. **Zenith Oral:** "Zenith" means the highest point, suggesting premium quality and reaching for the best in eco-friendly dental care.
13. **VeroBrush:** "Vero" means "truth" in Italian/Latin, implying authenticity and transparency.
14. **Aura Oral:** "Aura" suggests a distinctive, positive atmosphere, tying into natural wellness.
15. **Flow Oral:** Implies natural processes, ease, and a smooth experience.

III. Benefit-Oriented & Mission-Driven: (Focus on the positive impact and values)

16. **Conscious Clean:** Highlights the mindful choice consumers make for hygiene and the environment.
17. **Smile & Sustain:** Directly links personal well-being with planetary well-being.
18. **Planet Brush Co.:** Clear, direct, and emphasizes the global mission.
19. **Gentle Green Oral:** Focuses on being gentle on teeth and the planet.
20. **The Renew Brush:** Emphasizes renewable resources and a fresh start for dental care.
21. **Zero Grin:** A clever play on "zero waste" and a happy smile.

Tips for Choosing:

* **Check Availability:** See if the name (and its domain name) is available.
* **Pronunciation & Spelling:** Is it easy to say and spell?
* **Memorability:** Is it catchy and easy to remember?
* **Target Audience:** Does it resonate with your ideal customers?
* **Future Proofing:** Does it allow for potential expansion into other eco-friendly products?
* **Brand Story:** Which name best tells the story you want your company to convey?

Ex:9 Prompting and prompt templates.

Problem Statement : Create a prompt with one input variable. Generate a suitable name for a company that makes a particular product (use any product name). Input is the name of the product.

```
# Import ChatGoogleGenerativeAI from Langchain's Google Generative AI module
# This class allows you to use Google Gemini models for natural lang generation.
from langchain_google_genai import ChatGoogleGenerativeAI

# Initialize the Gemini model with specific parameters.
# 'model'- Gemini version, 'temperature'- controls creativity -0= factual,
  1= highly creative
llm = ChatGoogleGenerativeAI(model="gemini-2.5-flash", temperature=0.9)
product = "eco-friendly bamboo toothbrushes"
# Define the product for which you want a company name.
prompt = f"Suggest a creative and suitable name for a company
that makes {product}."

# Send the prompt to the Gemini model and get a response
response = llm.invoke(prompt)

# Print the AI-generated name to the console.
print(response.content)
```

```
In [1]: %runfile D:/GEN_AI_LAB/LAB2/EXP2.8.py --wdir
None of PyTorch, TensorFlow >= 2.0, or Flax have been found. Models won't be available and only tokenizers, configuration
and file/data utilities can be used.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760795175.602054   26872 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
Here are several suitable names for a company that makes handmade leather shoes, presented in Italian, with explanations
for their meaning and why they might be a good fit:

---

**Classic & Craft-Oriented:**

1.  **Maestri Calzolai**
    *   **Meaning:** "Master Shoemakers"
    *   **Why it works:** Directly conveys expertise, tradition, and the skill involved in crafting shoes. It evokes a
sense of heritage and quality.

2.  **Bottega del Cuoio**
    *   **Meaning:** "Leather Workshop" or "Leather Artisan's Shop"
    *   **Why it works:** Emphasizes the craftsmanship and specialization in leather goods. It suggests a place where
skilled artisans work with leather.

3.  **Il Calzificio**
    *   **Meaning:** "The Shoe Factory" (with a connotation of high-quality, artisanal production)
    *   **Why it works:** While "Calzificio" can imply a factory, it suggests a place where shoes are made with care
than just a mass-produced item.

4.  **Forma e Cuoio**
    *   **Meaning:** "Form and Leather"
    *   **Why it works:** Highlights the importance of shape and fit, combined with the material. It's concise and
memorable.

5.  **Manufatto Italiano**
    *   **Meaning:** "Italian Handcraft"
    *   **Why it works:** Emphasizes the handmade nature and the Italian origin, which is a strong selling point.
"Artigi
```

```
6.  **Il Passo Italiano**
    *   **Meaning:** "The Italian Step" or "The Italian Pace"
    *   **Why it works:** Evokes the elegance of Italian style, the journey, and the comfort of a well-made shoe. It's
poetic and hints at heritage.

7.  **Pelle Nobile**
    *   **Meaning:** "Noble Leather"
    *   **Why it works:** Emphasizes the high quality and luxurious nature of the leather used. "Nobile" suggests
exclusivity and superior material.

8.  **Anima di Pelle**
    *   **Meaning:** "Soul of Leather"
    *   **Why it works:** A more poetic and emotional name, suggesting that the shoes have character, depth, and are imbued
with the passion of their makers.

9.  **Stile su Misura**
    *   **Meaning:** "Custom Style" or "Tailored Style"
    *   **Why it works:** Highlights the bespoke nature, individual fit, and personalized elegance of handmade shoes.

10. **Eleganza Artigiana**
    *   **Meaning:** "Artisan Elegance"
    *   **Why it works:** Combines the beauty and sophistication of the shoes with the skilled, handmade process.

---

**Concise & Modern:**

11. **Punto Cuoio**
    *   **Meaning:** "Leather Stitch" or "Leather Point"
    *   **Why it works:** Simple, memorable, and refers to a key detail of handmade leatherwork - the precision of the
stitching.

12. **Calzatura Viva**
    *   **Meaning:** "Living Footwear" or "Vibrant Footwear"
    *   **Why it works:** Suggests comfort, durability, and shoes that are "alive" with quality and character, evolving
with the wearer.

---

When choosing, consider:

*   **Target Audience:** Who are you selling to?
*   **Brand Personality:** Do you want to emphasize tradition, luxury, innovation, or a blend?
*   **Pronunciation:** Is it easy for your target audience (even non-Italian speakers) to say and remember?
```

Ex10: Creating prompts with multiple input variables

Problem Statement : Generate a suitable name for a company that makes a particular product (use any product name) in a specific language. Inputs are the product and the language.

```
# Allows you to use Google gemini models for natural language generation
from langchain_google_genai import ChatGoogleGenerativeAI

#model -> Gemini version, 'temperature' - controls creativity - 0 = factual,
1 - highly creative
llm = ChatGoogleGenerativeAI (model = "gemini-2.5-flash",
    temperature = 0.9)

# Define the product for which you want a company name
product = "eco-friendly ~~bo~~ "handmade leather shoes"
language = "Italian"

# f-string allow variable values to be inserted directly to string
prompt = (f"Suggest a suitable name for a company that makes
  {product}."
  f" Please provide the name in {language}.")

# Send the constructed prompt to the Gemini model
response = llm.invoke (prompt)

# Print the generated company name
print (response.content)
```

```
In [1]: %runfile D:/GEN_AI_LAB/LAB4/EXP4.1.py --wdir
D:\ANACONDA\ANAKONDA SOFTWARE\Lib\site-packages\vertexai\_model_garden\_model_garden_models.py:278: UserWarning: This
feature is deprecated as of June 24, 2025 and will be removed on June 24, 2026. For details, see https://cloud.google.com/
vertex-ai/generative-ai/docs/deprecations/genai-vertexai-sdk.
  warning_logs.show_deprecation_warning()
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760805996.710743   22952 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
E0000 00:00:1760805999.037645   22952 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
E0000 00:00:1760805999.042069   22952 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
Generating image with Imagen, this may take a moment...
D:\ANACONDA\ANAKONDA SOFTWARE\Lib\site-packages\vertexai\vision_models\_vision_models.py:1436: UserWarning: This feature is
deprecated as of June 24, 2025 and will be removed on June 24, 2026. For details, see https://cloud.google.com/vertex-ai/
generative-ai/docs/deprecations/genai-vertexai-sdk.
  warning_logs.show_deprecation_warning()
D:\ANACONDA\ANAKONDA SOFTWARE\Lib\site-packages\vertexai\vision_models\_vision_models.py:153: UserWarning: This feature is
deprecated as of June 24, 2025 and will be removed on June 24, 2026. For details, see https://cloud.google.com/vertex-ai/
generative-ai/docs/deprecations/genai-vertexai-sdk.
  warning_logs.show_deprecation_warning()
Success! Image generated and saved to 'output_image.png'
You can find the file in the same folder as your script.
```

Date

Expt. No.

Page No.

Aim: To demonstrate text-to-image generation using Google Cloud Vertex AI's Imagen model.

Generate an image from the prompt using Google's Imagen model.

```
#Import the Google Cloud AI platform lib for Vertex AI interactions.
from google.cloud import aiplatform
#Import Vertex AI preview module for experimental or new features
import vertexai.preview
#Import the main Vertex AI module for initializing & using models
import vertexai
#Import ImageGenerationModel class to generate images from text prompts.
from vertexai.vision_models import ImageGenerationModel

#Set your Google Cloud-Project ID
PROJECT_ID = "virtual-firefly-474705-d8"
#A common location is "us-central1"
LOCATION = "us-central1"

#Initialize Vertex AI with your project ID and location
vertexai.init(project = PROJECT_ID, location = LOCATION)

#Load the pre-trained Imagen model for text-to-image generation
model = ImageGenerationModel.from_pretrained("imagegeneration@006")

#Define the text prompt describing the image to generate
prompt = "A watercolor image of Zurich with colorful reflections
in the water"
```

Teacher's Signature

Date

Expt. No.

Page No.

```
# Inform the user that the image generation process has started
print("Generating image with Imagen, this may take a moment...")

# Generate one image based on the prompt using the Imagen model
images = model.generate_images(prompt = prompt, number_of_images=1)

# Specify the file name to save the generated image
output_file = "output_image.png"

# Save the first generated image to the specified file
images[0].save(location=output_file)

# Inform the user that the image generation and saving were successful
print(f"Success! Image generated and saved to '{output_file}'")

# Remind the user where to find the saved image file
print("You can find the file in the same folder as your script.")
```

Teacher's Signature

Date

Expt. No.

Page No.

Generate Descriptions of a given image (image - to - text generation)

input : URL of an image

```
import google.generativeai as genai # Google GenAI python SDK
import os # to access environment variables like API keys
import requests # to fetch image data from a given URL
from PIL import Image # to handle and open image files
import io # to convert byte data into file-like objects

# Configure with your API key
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])

# Use a Gemini model that can handle vision
model = genai.GenerativeModel('gemini-2.5-flash')

# The URL of the image you want to describe
image_url = "https://images.....

print("Fetching image from URL")

# 1. Fetch the image data from the URL
try: # Begin error-handling block to manage network issues
    response = requests.get(image_url) # Send a GET req to download img
    response.raise_for_status() # Raise an error if HTTP req failed

# 2. Open the image from the downloaded bytes.
img = Image.open(io.BytesIO(response.content))
# Convert the downloaded bytes into an image object.
```

Teacher's Signature

```
In [1]: %runfile D:/GEN_AI_LAB/LAB4/EXP4.2.py --wdir
Fetching image from URL...
Image fetched. Asking Gemini to describe it...
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760806758.725060   20672 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.

--- GEMINI'S DESCRIPTION ---
This is a breathtaking low-angle shot of a vibrant, sun-drenched field of tulips, looking up towards a brilliant blue sky
and distant mountains.

Here's a detailed description:

*   **Sky and Sun:** The upper half of the image is dominated by a clear, deep cerulean blue sky. A radiant sunburst is
prominently positioned in the upper right quadrant, casting strong, bright rays that illuminate the scene. A few scattered,
fluffy white clouds drift across the sky, particularly on the right side.
*   **Background Landscape:** In the mid-ground, beneath the sky and behind the closer tulips, majestic, dark silhouettes
of mountains rise. Their peaks are somewhat hazy, and some lighter patches, possibly snow or rock, can be discerned. Lush
green trees and foliage are visible at the base of the mountains.
*   **The Tulip Field (Foreground and Mid-ground):** The image is filled with a dense, sprawling field of tulips, viewed
from a very low perspective, making their stems appear tall and reaching.
    *   **Colors:** There's a spectacular riot of colors, including vibrant fuchsia and magenta pinks, sunny yellows, vivid
oranges, and deep crimson reds. The tulips are scattered throughout the frame, creating a rich tapestry of hues.
    *   **Lighting:** The intense sunlight, coming from behind the flowers, backlights many of the tulip petals, making
them appear translucent and luminous. This effect highlights the delicate veins and inner structure of the petals, giving
them a glowing quality. Shadows are cast by the petals onto themselves and by the flowers onto the lush green leaves below.
    *   **Foliage:** The lower portion of the image, especially, showcases the broad, vibrant green leaves of the tulip
plants. They are densely packed, with the edges and some surfaces catching the sunlight, creating bright highlights against
deeper shadowed areas, giving them a rich, textural quality. The tall, slender green stems support the colorful blooms.
*   **Composition:** The low vantage point draws the viewer's eye upwards through the flowers towards the expansive sky and
sun, creating a dynamic and immersive feeling, as if one is standing amidst the tulips themselves.

Overall, the image evokes a sense of joyous abundance, the vibrant energy of spring, and the beauty of a bright, sunny day
in a picturesque natural setting.
```

Date

Expt. No.

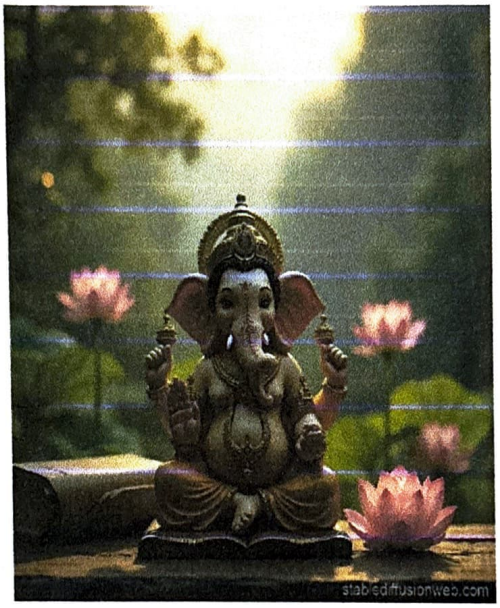
Page No.

```
print ("Image fetched. Asking Gemini to describe it...")

#3. Send both the text prompt and the image to the model.
response=model.generate_content(["What's in this image?",img])

#4. Print the text description
print ("\n--- GEMINI'S DESCRIPTION ---")
print (response.text) # generates description of the image
# Catch and handle any network-related errors
except requests.exceptions.RequestException as e:
    print (f"Error fetching image from URL: {e}")
    # print the error message if fetching fails
```

Teacher's Signature


stablediffusionweb.com

Date

Expt. No.

Page No.

Image-to-text generation for a custom image
input : path to the custom image

```
import google.generativeai as genai # Google GenAI SDK for Gemini access
import os # to access environment variables
from PIL import Image # PIL lib to open and process img files

#Configure with your API key
genai.configure(api_key=os.environ["GOOGLE_API_KEY"])

# Use a gemini model that can handle version
model = genai.GenerativeModel('gemini-2.5-flash')

# Define the file path to the image for description generation
img_path = "D:/GENAI_LAB.....jpg"

try: # Begin try-except block to handle errors during file reading
    print(f"Opening image from : {img_path}")
# 1. Open the local img file
    img = Image.open(img_path)# Load image from path
    print("Image opened. Asking Gemini to describe it...")

# 2. Send both the text prompt and the img object to the model
    response = model.generate_content(["What's in this image?", img])

# 3 Print the text description
    print("\n-- GEMINI'S DESCRIPTION")
    print(respone.text)
```

Teacher's Signature

```
In [1]: %runfile D:/GEN_AI_LAB/LAB4/EXP4.3.py --wdir
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
E0000 00:00:1760807744.164050   23716 alts_credentials.cc:93] ALTS creds ignored. Not running on GCP and untrusted ALTS is
not enabled.
Opening image from: D:/GEN_AI_LAB/LAB4/GANESHA.jpg
Image opened. Asking Gemini to describe it...

--- GEMINI'S DESCRIPTION ---
This image features a beautifully crafted statue of **Lord Ganesha** as the central subject.

Here's a detailed breakdown of what's in the image:

1.  **Lord Ganesha Statue:**
    *   A ceramic or resin statue of Ganesha, easily recognizable by his elephant head, large ears, and curled trunk.
    *   He is depicted seated in a meditative or relaxed posture, wearing an orange draped garment (dhoti).
    *   He is adorned with elaborate golden jewelry, including a crown with a prominent red jewel, multiple necklaces,
armlets, and bracelets.
    *   Red markings (tilak) are visible on his forehead and trunk.
    *   His hands are raised, one in a gesture of blessing or fearlessness (Abhaya Mudra), and the other appears to be in a
similar gesture or holding a small object.
    *   The statue rests on a decorative base, which itself resembles a stylized open book.

2.  **Lotus Flowers:**
    *   Several vibrant pink lotus flowers are present. One is prominently placed in the immediate foreground on the right,
fully bloomed.
    *   Others are visible in the mid-ground, slightly out of focus, behind and to the sides of Ganesha.
    *   Large green lotus pads (leaves) are also visible, suggesting a water setting or a garden environment.

3.  **Book:**
    *   To the left of Ganesha, in the foreground, there's a thick, old-looking book with light-colored pages and a rustic,
possibly leather-bound or parchment-like cover.

4.  **Setting:**
    *   The statue, book, and foreground lotus are placed on a textured, dark wooden surface, showing distinct grain.
    *   The background is soft and out-of-focus, depicting lush green foliage, suggesting a natural outdoor environment
like a garden, forest, or tranquil temple grounds.
    *   A warm, hazy golden light illuminates the upper center background, creating a serene and ethereal atmosphere,
possibly indicating sunrise or sunset.

The overall impression is one of peace, spirituality, and reverence for the deity, set in a tranquil natural environment.
```

Date

Expt. No.

Page No.

```
except FileNotFoundError : # Handle case when the given img path
 is incorrect or file missing
    print (f"Error : The file was not found at the path : {img_path}")
except Exception as e : # Catch & handle any other unexpected exceptions
    print (f"An error occurred : {e}") # error details for debugging
```

Teacher's Signature